Behind the scenes of Rust

Alex Crichton
Announcing Rust 1.0

May 15, 2015 • The Rust Core Team

Today we are very proud to announce the 1.0 release of Rust, a new programming language aiming to make it easier to build reliable, efficient systems. **Rust combines low-level control over performance with high-level convenience and safety guarantees.** Better yet, it achieves these goals without requiring a garbage collector or runtime, making it possible to use Rust libraries as a “drop-in replacement” for C. If you’d like to experiment with Rust, the “Getting Started” section of the Rust book is your best bet (if you prefer to use an e-reader, Pascal Hertleif maintains unofficial e-book versions as well).

What makes Rust different from other languages is its type system, which represents a refinement and codification of “best practices” that have been hammered out by generations of C and C++ programmers. As such, Rust has something to offer for both experienced systems programmers and newcomers alike: experienced programmers will find they save time they would have spent debugging whereas newcomers can write low-level code without worrying about minor mistakes leading to mysterious crashes.

What does it mean for Rust to be 1.0?
Tooling

Infrastructure

Community
We’re not in 1970

• Developers have high expectations for tooling support

• Developers also expect a lot of tools

• Tools alone can often spawn entire ecosystems
Tools in Rust

- cargo
- rustdoc
- rustup
- rustc
- gdb
- RLS
- rustfmt
Why Cargo?
Why Cargo?

• Sharing code is *critical* for a young language

• A package manager from day one is *hugely* beneficial

• Despite blocking 1.0 on Cargo, Cargo accelerated stabilization of libstd through aggressive pruning
How has Cargo helped?

- Small standard library far easier to port and maintain
- “Dependency hell” is almost a thing of the past
- Language features like custom derive far easier with Cargo
What else was missing?

Immaturity of the Tooling

Another strong theme for improvement was the relative immaturity of the tooling for Rust. While tools like Cargo have been invaluable to a number of Rust users, other tools need attention.

Of non-Rust users, 1 in 4 responded that they aren’t currently using Rust because of the lack of strong IDE support. As one user puts it “[f]or a complex language like Rust, good editor tooling makes the learning process interactive.” Modern IDEs have become a powerful way to explore unfamiliar APIs, unfamiliar language features, and unfamiliar error messages.
Enter the LSP

**Language Server Protocol**

- User opens document
  - textDocument/didOpen: textDocument

- User edits document
  - textDocument/didChange: textDocument

- Language server sends errors/warnings
  - textDocument/publishDiagnostics: diagnostics

- User executes “Go To Definition”
  - textDocument/definition: textDocument, position
    - result: uri, range

- User closes document
  - textDocument/didClose: textDocument

**JSON RPC**
Enter the RLS

Rust Language Server (IDE support) #1317

Merged

alexcrichton merged 6 commits into rust-lang:master from nrc:ide on Feb 10, 2016

Conversation 237  Commits 6 Files changed 1

nrc commented on Oct 12, 2015

This RFC describes how we intend to modify the compiler to support IDEs. The intention is that support will be as generic as possible. A follow-up internals post will describe how we intend to focus our energies and deploy Rust support in actual IDEs.

There are two sets of technical changes proposed in this RFC: changes to how we compile, and the creation of an 'oracle' tool (name of tool TBC).

Thanks to Phil Dawes, Bruno Medeiros, Vosen, eddyb, Evgeny Kurbatsky, and Dmitry Jemerov for early feedback.

16

Changes to the compiler to support IDEs

nrc added T-tools T-compiler labels on Oct 12, 2015
RLS requirements

- Needs to support common IDE queries like
  - go to definition
  - find all uses
  - renaming variables
- Can’t reimplement all of rustc
- Responses must be fast
Getting rustc fast

- Parsing
- Macro Expansion
- Analysis
- Optimization
- Codegen
- Linking
Getting `rustc` fast

Where's the definition of this type?
Polishing the RLS

rustup component add rls-preview
Tools in Rust

- cargo
- rustdoc
- rustup
- rustc
- gdb
- RLS
- rustfmt
Tooling
Infrastructure
Community
We’re still not in 1970

- Developers expect their tools to not break
- Developers expect low friction when managing tools
- Vast majority of users will always be new ones
Infrastructure of Rust

- Continuous Integration
- `{www,doc,play}.rust-lang.org`
- crates.io
- Rapid release cycle
- “Dealing with GitHub”
- AWS services, CDNs, storage, etc
Rust is a systems programming language that runs blazingly fast, prevents segfaults, and guarantees thread safety.

See who's using Rust.

- Managed via PRs on GitHub
- Continuously deployed
- Delivered via CloudFront CDN
- Fun with DNS/SSL/…
use std::f64;

#[no_mangle]
pub fn foo() -> f64 {
    if bar(f64::INFINITY) as i8 != i8::max_value() {
        unsafe { exit(); }
    }
    bar(f64::NEG_INFINITY)
rustup is an installer for
the systems programming language **Rust**

Run the following in your terminal, then follow
the onscreen instructions.

```
curl https://sh.rustup.rs -sSf | sh
```


* rustup is an official Rust project.
  * [other installation options](https://doc.rust-lang.org/rustup/index.html)
  * [about rustup](https://doc.rust-lang.org/rustup/build-guide.html)
rustup update

- Also delivered via CDN
- Delivers over 50 targets to compile to
- Crazy platform-specific logic in rustup itself
cargo build

- crates.io crates delivered via CDN
- crates.io itself written in Rust
- Deployed via Heroku
You’ve found a bug!

- All rustc/Rust language development happens on GitHub
- GitHub is what most know and love, makes it easiest for new contributors
- Let’s send a PR…
No description provided.

Thanks for the pull request, and welcome! The Rust team is excited to review your changes, and you should hear from @eddyb (or someone else) soon.

If any changes to this PR are deemed necessary, please add them as extra commits. This ensures that the reviewer can see what has changed since they last reviewed the code. Due to the way GitHub handles out-of-date commits, this should also make it reasonably obvious what issues have or haven't been addressed. Large or tricky changes may require several passes of review and changes.

Please see the contribution instructions for more information.
arielb1 commented 3 hours ago

@bors r+
bors: Mandatory homage
The Last of the Masters

The title character, Bors, a 200-year-old "government integration robot" — and the last in existence — awakens after a routine maintenance check to learn that his motor system is in a state of decline. An artificially intelligent machine who displays a degree of emotion and even psychological complexity, he is informed by Fowler, a personal mechanic, that his body has begun to break down due to age.
bors never sleeps

37,864 contributions in the last year

Learn how we count contributions.
bors in action

eddyb commented 3 days ago

@bors r+

bors commented 3 days ago

⚠️ Commit 27d5872 has been approved by eddyb
**bors in action**
bors in action

**bors** commented 3 days ago

Commit **27d5872** has been approved by **eddyb**

**kennytm** added the **S-waiting-on-bors** label 3 days ago

**bors** commented 5 hours ago

Testing commit **27d5872** with merge **0400312**...

**bors** added a commit that referenced this pull request 5 hours ago

Auto merge of #46203 - nikomatsakis:type-foldable-macro, r=eddyb...
bors in action

Testing commit 27d5872 with merge 0400312 ...

bors added a commit that referenced this pull request 5 hours ago

Auto merge of #46203 – nkomatsakis/tpe-foldable-macro, r=eddyb ...

bors commented 3 hours ago

Test successful - status-appveyor, status-travis
Approved by: eddyb
Pushing 0400312 to master...

bors merged commit 27d5872 into rust-lang:master 3 hours ago
2 checks passed
Auto merge of #46203 - nikomatsakis: type-foldable-macro, r=eddyb

introduce macros for type-foldable and lift, convert stuff to use them

A random commit from a branch I've shelved for the time being that made
`TypeFoldable` stuff a bit less annoying to write.

r? @eddyb

<table>
<thead>
<tr>
<th>JOB NAME</th>
<th>TESTS</th>
<th>DURATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Environment: MSYS_BITS=64, RUST_CONFIGURE_ARGS=--build=x86_64-pc-windows-msvc --enable-profiler, SCRIPT=</td>
<td>1 hr 58 min</td>
<td></td>
</tr>
<tr>
<td>Environment: MSYS_BITS=32, RUST_CONFIGURE_ARGS=--build=i686-pc-windows-msvc --target=i686-pc-windows-msvc --enable-profiler, SCRIPT</td>
<td>2 hr 27 min</td>
<td></td>
</tr>
<tr>
<td>Environment: MSYS_BITS=64, RUST_CHECK_TARGET=check-aux, RUST_CONFIGURE_ARGS=--build=x86_64-pc-windows-msvc --enable-profiler, SCRIPT</td>
<td>2 hr 7 min</td>
<td></td>
</tr>
<tr>
<td>Environment: MSYS_BITS=64, SCRIPT=python x.py test src/tools/cargotest, RUST_CONFIGURE_ARGS=--build=x86_64-pc-windows-msvc --enable-profiler, SCRIPT=</td>
<td>1 hr 26 min</td>
<td></td>
</tr>
<tr>
<td>Environment: MSYS_BITS=32, RUST_CONFIGURE_ARGS=--build=i686-pc-windows-gnu, SCRIPT=python x.py test, MSYS_BITS=32, SCRIPT=</td>
<td>1 hr 29 min</td>
<td></td>
</tr>
<tr>
<td>Environment: MSYS_BITS=64, SCRIPT=python x.py test, RUST_CONFIGURE_ARGS=--build=x86_64-pc-windows-gnu,</td>
<td>2 hr 1 min</td>
<td></td>
</tr>
<tr>
<td>Environment: RUST_CONFIGURE_ARGS=--build=x86_64-pc-windows-msvc --enable-extended --enable-profiler, SCRIPT=</td>
<td>1 hr 29 min</td>
<td></td>
</tr>
<tr>
<td>Environment: RUST_CONFIGURE_ARGS=--build=i686-pc-windows-msvc --target=i686-pc-windows-msvc --enable-extension, SCRIPT=</td>
<td>1 hr 32 min</td>
<td></td>
</tr>
<tr>
<td>Environment: MSYS_BITS=32, RUST_CONFIGURE_ARGS=--build=i686-pc-windows-gnu --enable-extended, SCRIPT=</td>
<td>1 hr 11 min</td>
<td></td>
</tr>
<tr>
<td>Environment: MSYS_BITS=64, SCRIPT=python x.py dist, RUST_CONFIGURE_ARGS=--build=x86_64-pc-windows-gnu --enable-extended, SCRIPT</td>
<td>1 hr 33 min</td>
<td></td>
</tr>
<tr>
<td>Environment: MSYS_BITS=64, RUST_CONFIGURE_ARGS=--build=x86_64-pc-windows-msvc --enable-extended --enable-extended</td>
<td>1 hr 40 min</td>
<td></td>
</tr>
</tbody>
</table>
rust-lang / rust

auto  Auto merge of #46203 - introduce macros for type to use them

Commit 0400312
Compare 2ca00a9..0400312
Branch auto

bors authored and committed

Build Jobs

- #63241.1
- #63241.2
- #63241.3
- #63241.4
- #63241.5
- #63241.6
- #63241.7
- #63241.8
- #63241.9
- #63241.10
Release trains

- Nightly, beta, stable channels
- Automatic nightlies each night
- Stable/beta updated once every 6 weeks
- Stable/beta receive bug fixes with “backports”
Infrastructure of Rust

- Continuous Integration
- `{www,doc,play}.rust-lang.org`
- crates.io
- Rapid release cycle
- “Dealing with GitHub”
- AWS services, CDNs, storage, etc
Tooling

Infrastructure

Community
We get it, it’s not 1970

- Development does not happen in isolation any more
- Everyone’s got thoughts (often great ones!)
- Early stage projects live and die by their communities
Rust’s Community

- Governance of Rust itself
- RFC process
- Internals/users forum, IRC
- Community team
- Conferences
Rust RFCs

Many changes, including bug fixes and documentation improvements can be implemented and reviewed via the normal GitHub pull request workflow.

Some changes though are "substantial", and we ask that these be put through a bit of a design process and produce a consensus among the Rust community and the sub-teams.

The "RFC" (request for comments) process is intended to provide a consistent and controlled path for new features to enter the language and standard libraries, so that all stakeholders can be confident about the direction the language is evolving in.

Table of Contents

- Opening
- Table of Contents
- When you need to follow this process
- Before creating an RFC
- What the process is
- The RFC life-cycle
- Reviewing RFCs
- Implementing an RFC
- RFC Postponement
- Help this is all too informal!
To *int* or not to *int*?

RFC: Rename `int/uint` to something better #544

*merged* aturon merged 20 commits into *rust-lang:master* from *CloudiDust:int-to-intx* on Jan 6, 2015

CloudiDust commented on Dec 28, 2014

This RFC proposes that we rename the pointer-sized integer types *int/uint*, so as to avoid misconceptions and misuses.

This is yet another attempt to rename *int/uint*. See *A tale of two's complement* for reasons of the rejection of the previous proposal.

After community discussions, this RFC has undergone several major revisions and the originally proposed *intx/uintx* have lost favour.

The winners are: *isize/usize*!
Reducing entropy

RFC: Scaling Rust's Governance #1068

alexcrichton merged 3 commits into rust-lang:master from aturon:rust-governance on May 7, 2015

This RFC proposes to expand, and make more explicit, Rust's governance structure. It seeks to supplement today's core team with several subteams that are more narrowly focused on specific areas of interest.

Thanks to Nick Cameron, Manish Goregaokar, Yehuda Katz, Niko Matsakis and Dave Herman for many suggestions and discussions along the way.
The Rust Team

The Rust project is governed by a number of teams, each focused on a specific area of concern. Below are the rosters, in alphabetical order.

To contact a team, post your question or comment to the Internals forum and tag your post with the category corresponding to the team name. Note that security disclosures should follow the Rust security disclosure process.

Core team

Responsibility: overall direction of the project, subteam leadership, cross-cutting concerns

Nick Cameron  Alex Crichton  Yehuda Katz  Steve Klabnik

Niko Matsakis  Carol Nichols  Aaron Turon  Erick Tryzelaar
impl Future for Rust

Sep 18, 2017 • Aaron Turon

The Rust community has been hard at work on our 2017 roadmap, but as we come up on the final quarter of the year, we’re going to kick it into high gear—and we want you to join us!

Our goals for the year are ambitious:

- Rust should have a lower learning curve.
- Rust should have a pleasant edit-compile-debug cycle.
- Rust should provide a solid, but basic IDE experience.
- Rust should provide easy access to high quality crates.
- Rust should be well-equipped for writing robust, high-scale servers.
- Rust should have 1.0-level crates for essential tasks.
- Rust should integrate easily into large build systems.
- Rust’s community should provide mentoring at all levels

To finish off these goals, we intend to spend the rest of the year focused purely on “implementation” work—which doesn’t just mean code! In particular, we are effectively spinning down the RFC process for 2017, after having merged almost 90 RFCs this year!

So here’s the plan. Each Rust team has put together several working groups focused on a specific sub-area. Each WG has a leader who is responsible for carving out and coordinating work, and a dedicated